COMP90042

Web search and text analysis

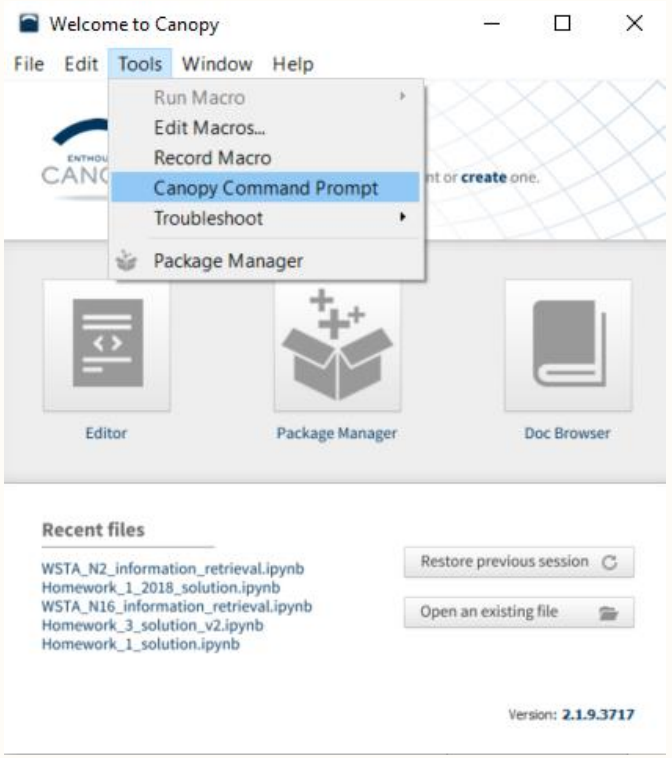
Workshop Week 2

Your tutor

- Winn Chow (Senior Tutor)
- winn.chow1@unimelb.edu.au
- Office: Doug McDonell - 9.23
- Here, you can find my workshop slides:
- https://github.com/winnchow/COMP90042-Workshops

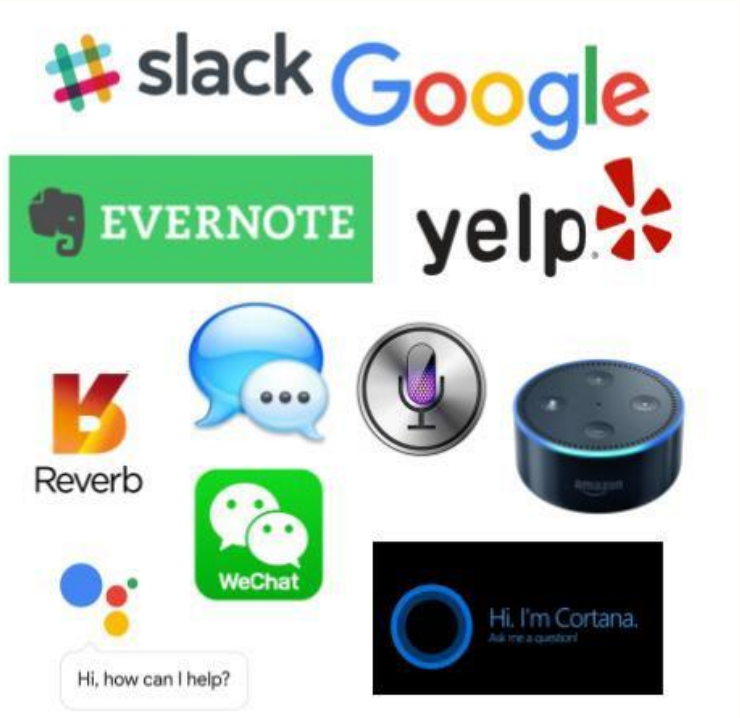
Python 3

- Familiarise yourself with Python 3
- https://trevorcohn.github.io/comp90042/workshops/week1-python-01.pdf
- Canopy: https://store.enthought.com/downloads/
- NLTK: https://www.nltk.org/install.html using Canopy Command Prompt



Text Processing

- Natural Language Processing (NLP)
- Language-aware applications are ones that can:
- "Leverage natural language processing techniques to understand human-generated text and audio data… [and] curate the myriad of human-generated information on the web specifically on our behalf, offering new and personalized mechanisms of human-computer interaction" (Bengfort, Bilbro and Ojeda, 2016).



More and more companies are applying natural language processing applications to everyday business problems.

Source: https://medium.com/@bytecubed/natural-language-processing-for-everyday-people-d6d0e4baf313

Google Duplex

- A.I. Assistant Calls Local Businesses To Make Appointments
- https://youtu.be/D5VN56jQMWM?t=59

Tokenization

- http://blog.xnextcon.com/?p=233

```
from nltk.tokenize import word_tokenize
```

```
sentence = "Hello Aswathi How are you doing today"
sentence_token = word_tokenize(sentence)
sentence_token
```

```
['Hello', 'Aswathi', 'How', 'are', 'you', 'doing', 'today']
```

Tokenisation

- https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html
- A *token* is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing.
- A *type* is the class of all tokens containing the same character sequence.
- E.g. he says he he

Stemming and Lemmatisation

- http://blog.xnextcon.com/?p=233

```
from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()
```

```
word = "mulitplying"
stem.stem(word)
```

```
'mulitpli'
```

```
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()
```

```
word = "multiplying"
lem.lemmatize(word,"v")
```

```
'multiply'
```

- Which one is simpler? Which one will give you back a valid word (using a lexicon e.g. a dictionary)?

Stemming and Lemmatisation

- In linguistics, morphology is the study of words.
- Inflectional morphology
- https://semanticsmorphology.weebly.com/inflectional-and-derivational-morphemes.html
  - Conform to grammatical constraints
  - Do not really alter the meaning
  - Both stemming and lemmatization

can handle it

English Inflectional Morphemes		Added to	Examples
-s	plural	*Nouns*	*She has got two guitar**s**.*
-'s	possessive	*Nouns*	*Zeynep**'s** hair is long.*
-er	comparative	*Adjectives*	*Zeynep has long**er** hair than Derya.*
-est	superlative	*Adjectives*	*Zeynep has the long**est** hair.*
-s	3rd person singular present tense	*Verbs*	*Zeynep play**s** the guitar.*
-ed	past tense	*Verbs*	*She play**ed** the guitar at the party.*
-ing	progressive	*Verbs*	*She is play**ing** the guitar at the party*
-en	past participle	*Verbs*	*She has tak**en** the guitar to the party.*

Stemming and Lemmatisation

Some English derivational affixes

Affix	Change	Examples
Suffixes		
-able	V → A	fix-able, do-able
-(at)ion	V → N	realiz-ation
-ing	V → N	the shoot-ing, the danc-ing
-ing	V → A	the sleep-ing giant
-ive	V → A	assert-ive
-al	V → N	refusal
-ment	V → N	treat-ment
-ful	N → A	hope-ful

- Derivational morphology
- https://www.slideshare.net/FirraBannie/morphology-derivation
- One class to another e.g. Verb (teach) -> Noun (teacher)
- Alter the meaning
- Stemming?
- Lemmatisation?

Term-document matrix

Query: **tea me**

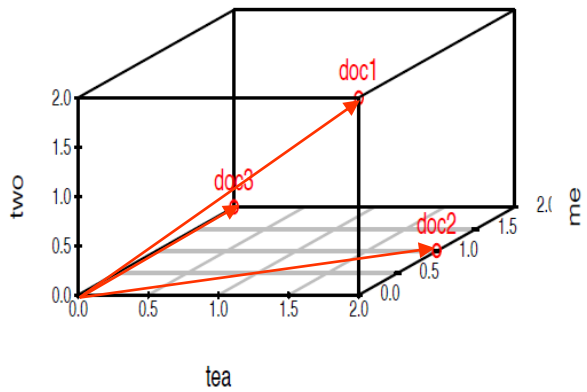
doc1	Two for tea and tea for two
doc2	Tea for me and tea for you
doc3	You for me and me for you

How about?

Query: two

	two	tea	me	you
doc1	2	2	0	0
doc2	0	2	1	1
doc3	0	0	2	2

Term Space



Inverted index

Query: **tea me**

doc1	Two for tea and tea for two
doc2	Tea for me and tea for you
doc3	You for me and me for you

Term		Postings list
tea	→	1:1.4 ; 3:1.0 ; 6:1.7 ; . . .
two	→	2:2.3 ; 3:1.0 ; 4:1.7 ; . . .
me	→	1:1.0 ; 2:1.4 ; . . .

- weights listed are the normalised TF*IDF values

$$tf_{d,t} \times idf_t$$

TF*IDF similarity score for a term

Distribution of terms in a document:

- $tf_{d,t}$ = term frequency of a document (count of a term *t* in a document *d*)
- ***per term per document***

How specific is the term? Does it appear only in a few documents?

- df_t = document frequency of a term (count of documents that contain the term *t*)
- Idf_t = inverse document frequency (the *fewer* the documents the better. It must be a *very specific term*.)
- *N* = total number of documents
- ***per term***

$$idf_t = \log \frac{N}{df_t}$$

TF*IDF similarity score for a Query

- Compute for a query Q and for each document that contains at least one of the terms in the query

$$S_{\text{TF-IDF}}(d, Q) = \sum_{t \in Q} tf_{d,t} \times \log \frac{N}{df_t}$$

$$S_{\text{TF-IDF}}(d, Q) = \sum_{t \in Q} tf_{d,t} \times \log \frac{N}{df_t}$$

Q4:
Query: “apple ibm”

$tf_{d,t}$

	apple	ibm	lemon	sun
D_1	4	0	1	1
D_2	5	0	5	0
D_3	2	5	0	0
D_4	1	0	1	7
D_5	0	1	3	0

- N = ?
- df_t = ?

	apple	ibm	lemon	sun
idf	$\log \frac{5}{4} = 0.22$	$\log \frac{5}{2} = 0.92$	$\log \frac{5}{4} = 0.22$	$\log \frac{5}{2} = 0.92$

Q4:
TF*IDF similarity scores

	apple	ibm	lemon	sun
D_1	0.89	0	0.22	0.92
D_2	1.12	0	1.12	0
D_3	0.45	4.58	0	0
D_4	0.22	0	0.22	6.41
D_5	0	0.92	0.67	0

Okapi BM25

What happens?

- $k_1 = 0$ (binary model or term frequency)
- $k_3 = 0$ (binary model or term frequency)
- b = 1 (scaling by document length)

$$w_t = \log \frac{N - f_t + 0.5}{f_t + 0.5} \times \frac{(k_1 + 1) f_{d,t}}{k_1((1 - b) + b\frac{L_d}{L_{\text{avg}}}) + f_{d,t}} \times \frac{(k_3 + 1) f_{q,t}}{k_3 + f_{q,t}}$$

where f_t is the document frequency of term t, $f_{d,t}$ is the term frequency of term t in document d and $f_{q,t}$ is the term frequency of term t in query q. k_1, k_3 and b are parameters with $0 \leq k_1 < \infty$, $0 \leq k_3 < \infty$ and $0 \leq b \leq 1$. L_d is the length of document d and L_{avg} is the average document length in the collection.

- TF (Document)? TF (Query)?
- IDF (Document)?

Very Useful Online Resources

- https://web.stanford.edu/~jurafsky/
- 2012 NLP MOOC w/Chris Manning:
  - Youtube channel lecture videos
  - Slides